**Create the ErrorLog Table**

Create a table in your database to store error logs:

**✅ SQL Script:**

sql

CopyEdit

CREATE TABLE [dbo].[ErrorLog] (

[Id] INT IDENTITY(1,1) PRIMARY KEY,

[UserName] NVARCHAR(100) NULL,

[LogDate] DATETIME NOT NULL DEFAULT GETDATE(),

[ExceptionMessage] NVARCHAR(MAX) NOT NULL,

[StackTrace] NVARCHAR(MAX) NULL,

[Source] NVARCHAR(255) NULL,

[AdditionalInfo] NVARCHAR(MAX) NULL

);

**2. Create LoggerService**

Create a LoggerService class that uses **Dapper** to insert logs into the table.

**✅ LoggerService.cs**

1. Create a folder **Services** and add this file:

csharp

CopyEdit

using Dapper;

using System;

using System.Data;

using System.Threading.Tasks;

public class LoggerService

{

private readonly IDbConnection \_dbConnection;

public LoggerService(IDbConnection dbConnection)

{

\_dbConnection = dbConnection;

}

public async Task LogErrorAsync(Exception ex, string userName = null, string additionalInfo = null)

{

var sql = @"

INSERT INTO [ErrorLog]

(UserName, LogDate, ExceptionMessage, StackTrace, Source, AdditionalInfo)

VALUES

(@UserName, @LogDate, @ExceptionMessage, @StackTrace, @Source, @AdditionalInfo)";

var parameters = new

{

UserName = userName,

LogDate = DateTime.UtcNow,

ExceptionMessage = ex.Message,

StackTrace = ex.StackTrace,

Source = ex.Source,

AdditionalInfo = additionalInfo

};

try

{

await \_dbConnection.ExecuteAsync(sql, parameters);

}

catch (Exception dbEx)

{

// If logging fails, log to console or a file

Console.WriteLine($"Logging failed: {dbEx.Message}");

}

}

}

**3. Modify BaseController to Handle Errors**

* Modify your existing BaseController to inject LoggerService and handle logging inside catch blocks.

**✅ BaseController.cs**

1. Add error handling and logging in the BaseController:

csharp

CopyEdit

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.Extensions.Logging;

using Dapper;

using System;

using System.Data;

using System.Threading.Tasks;

namespace BCES.Controllers.Base

{

public class BaseController : Controller

{

private readonly IDbConnection \_dbConnection;

private readonly IHttpContextAccessor \_httpContextAccessor;

private readonly ILogger<BaseController> \_logger;

private readonly LoggerService \_loggerService;

public BaseController(

DapperContext dapper,

IHttpContextAccessor httpContextAccessor,

ILogger<BaseController> logger,

LoggerService loggerService)

{

\_dbConnection = dapper.CreateConnection();

\_httpContextAccessor = httpContextAccessor;

\_logger = logger;

\_loggerService = loggerService;

}

public override async Task OnActionExecutionAsync(ActionExecutingContext context, ActionExecutionDelegate next)

{

try

{

string userADId = \_httpContextAccessor.HttpContext.User.Identity.Name;

string[] temp = userADId?.Split('\\');

userADId = temp?.Length > 1 ? temp[1] : userADId;

var userName = \_dbConnection.QueryFirstOrDefault<string>(

@"SELECT u.UserName FROM SBCES.Users u WHERE u.UserADId = @UserADId",

new { UserADId = userADId }

);

if (string.IsNullOrEmpty(userName))

{

context.Result = View("~/Views/Shared/Unauthorized.cshtml");

return;

}

// Execute action

await next();

}

catch (Exception ex)

{

string userName = \_httpContextAccessor.HttpContext.User?.Identity?.Name;

// 1. Log using ILogger

\_logger.LogError(ex, "An error occurred in BaseController -> OnActionExecutionAsync");

// 2. Log to ErrorLog table using LoggerService

await \_loggerService.LogErrorAsync(ex, userName, "Error in BaseController -> OnActionExecutionAsync");

// Redirect to an error page

context.Result = View("~/Views/Shared/Error.cshtml");

}

}

}

}

**4. Register Dependencies in Program.cs**

You need to register LoggerService, IDbConnection, and other dependencies:

**✅ Program.cs**

1. Open Program.cs and register the services:

csharp

CopyEdit

var builder = WebApplication.CreateBuilder(args);

// Logging (Console + Debug)

builder.Logging.AddConsole();

builder.Logging.AddDebug();

// Database connection (Dapper)

builder.Services.AddScoped<IDbConnection>(sp =>

new SqlConnection(builder.Configuration.GetConnectionString("DefaultConnection")));

// Logger service

builder.Services.AddScoped<LoggerService>();

// HttpContextAccessor

builder.Services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Global exception handler

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error"); // Redirect to error page

}

app.UseRouting();

app.UseAuthorization();

app.MapControllers();

app.Run();

**5. Create an Error Page (Optional)**

You can create a simple error page to handle the error gracefully:

**✅ Views/Shared/Error.cshtml**

1. Create an error view:

html

CopyEdit

@{

ViewData["Title"] = "Error";

}

<h1 class="text-danger">Oops! Something went wrong.</h1>

<p>We're working to fix the issue. Please try again later.</p>

**6. How to Use in catch Blocks**

If you want to log an error manually in a controller, you can just call \_loggerService directly.

**✅ Example in BusesController**

1. Example usage in an inherited controller:

csharp

CopyEdit

public class BusesController : BaseController

{

private readonly IDbConnection \_dbConnection;

public BusesController(

DapperContext dapper,

IHttpContextAccessor httpContextAccessor,

ILogger<BaseController> logger,

LoggerService loggerService)

: base(dapper, httpContextAccessor, logger, loggerService)

{

\_dbConnection = dapper.CreateConnection();

}

public async Task<IActionResult> GetBusDetails(string rebuiltStockNum)

{

try

{

var query = @"

SELECT rlb.[RebuiltStockNum],

lb.[ListId],

lb.[Description]

FROM [SBCES].[RbListOfBuses] rlb

INNER JOIN [SBCES].[ListOfBuses] lb ON rlb.[ListId] = lb.[ListId]

WHERE rlb.[RebuiltStockNum] = @RebuiltStockNum";

var result = await \_dbConnection.QueryAsync<BusesModel>(query, new { RebuiltStockNum = rebuiltStockNum });

return Ok(result);

}

catch (Exception ex)

{

// Log directly in the catch block if needed

await \_loggerService.LogErrorAsync(ex, User.Identity?.Name, "Error in GetBusDetails");

return StatusCode(500, "An error occurred");

}

}

}